(IJCSIS) International Journal of Computer Science and Information Security,
Vol. 17, No. 5, May 2019

Botnet Detection and Prevention in Software Defined Networks (SDN) using DNS Protocol

Paper

By

Muhammad Junaid Zafar,Riphah International University, Islamabad, Pakistan
mjunaidzafar@hotmail.com,
Professor Dr. Muhammad Zubair, Riphah International University, Islamabad, Pakistan
m.zubair@riphah.edu.pk

Abstract

Software defined networks (SDNs) is one of the most emerging field and will cause revolution in the Information Technology (IT) industry. The flexibility in the SDNs make it most attractive technology to adopt in all type of networks. This flexibility in the network made the SDNs more prone to the security issues so it is important to cater these issues in start from the SDN design up-to the deployment and operations. This Paper proposed a DNS based approach to prevent SDNs from botnet by applying one million web database concept without reading packet payload. To do any activity, Bot need to communicate with CnC and requires DNS to IP resolution. For any request having destination port 53 (DNS) will be checked. The protocol will get all matching traffic and will send it to 1Mdb. If URL Exists in 1Mdb then do not respond otherwise send reply with remove flow and block flow to the controller. This approach will use Machine learning algorithms to classify the traffic as BOT or normal traffic. Naive Bayes Classifier is used to classify the data using python programming language. The selection of dataset is very important task for machine learning based botnet detection and prevention techniques. The poor selection of dataset possibly lead to biased results. The real world and publically available dataset is a good choice for evaluation of botnet detection techniques. To meet these criteria, publicly available CTU-43 botnet dataset has been used. This dataset provide packet dumps (pcap files) of seven real botnets (Neris, Rbot, Virut, Murlo, Menti, Sogou, and NSIS). We will use these files to generate botnet traffic for evaluation and test our model. To generate normal traffic, we selected ISOT dataset. This dataset provides a single pcap file having normal traffic and traffic for weladec and zeus botnet.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Contents

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

List of Figures

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

List of Tables

List of Abbreviations and Symbols

Abbreviations

CNC	Command and Control
SDN	Software Defined Networks
DNS	Domain Name System
1Mdb	One Million Database

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Chapter 1

Introduction

Software-defined networking (SDN) is an architecture that aims to make networks agile and flexible. The goal of SDN is to improve network control by enabling enterprises and service providers to respond quickly to changing business requirements. In a software-defined network, a network engineer or administrator can shape traffic from a centralized control console without having to touch individual switches in the network. The centralized SDN controller directs the switches to deliver network services wherever they're needed, regardless of the specific connections between a server and devices. A typical representation of SDN architecture comprises three layers: the application layer, the control layer and the infrastructure layer. The application layer, not surprisingly, contains the typical network applications or functions organizations use, which can include intrusion detection systems, load balancing or firewalls. Where a traditional network would use a specialized appliance, such as a firewall or load balancer, a software-defined network replaces the appliance with an application that uses the controller to manage data plane behavior. The control layer represents the centralized SDN controller software that acts as the brain of the software-defined network. This controller resides on a server and manages policies and the flow of traffic throughout the network.

This document covers and introduced a technique to secure the SDN from BOTs using a new method. The document covers the detail of software defined networks, BOTs, existing techniques used to detect and protect SDNs from BOTs, issues in existing techniques used to secure SDNs from BOTs and new technique introduced to secure SDNs from BOTs.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

1.1 Problem Area

Security is one of the main concern in Software Defined networks due to the centralized controlled nature of the solution. It is very important to secure the controller from all type of threats. One of the major threat for SDN is to protect the SDN and SDN Controllers from BOTs. If not secured and in case controller compromised then whole network will be compromised considering that controller is controlling the whole network.

1.2 Problem Statement

Security of SDN from BOTs and BOTNETs is very important. It is very important to detect the BOTs in very start of communication before the BOTs damage the network. In case BOTs successful in communication with the Command and Control Center (CnC) then it will get the execution command from the CnC and will cause security issues in the SDN and can take over the controller control as well to get the complete control of the network. It is very important to secure the SDNs from BOTs.

1.3 Research Questions

1. What are the limitations in existing BOTNET detection and prevention techniques.
2. What are the ways to stop the BOTs in start of communication before it damages the Software Defined Networks.
3. What are the ways to stop the BOT to contact with their CnCs.
4. Which protocol is the best to detect BOTNET communication.

1.4 Research Objectives

Software Defined Networks (SDNs) are effected by BOTs and damage the network once compromised by BOTs. Existing techniques can detect the BOTs in SDNs but have some limitations due to which limited or high impact damage to the network can occur. Objective of the research done in this Paper is to introduce a new technique for detection and prevention of BOTs in Software defined network using DNS protocol. The new technique have some benifits to overcome the limitations of existing techniques. This

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

will secure the SDNs completely from BOTs in very start of communication.

1.5 Significance of Work and Potential Benefits

The approach have a lot of benefits as compare to other approaches i.e. Very less computations on live traffic, Only Traffic for Port 53 will be monitored, No computation or processing on remaining traffic, Bots will be blocked at very start of communication, Infected IPs will be permanently blocked, System not only provide detection but also prevention solution, standard Technique for all type of controllers and protocols, isolate infected system before start of communication.

1.6 Novel Contributions

A detailed literature survey has been performed with near about 40 papers have been studied, 10 very focused on the topic and research questions are selected. Limitations of existing research has been addresses in the proposed methodology which will have a great contribution in the research area. The proposed methodology is completely different and new which has been never proposed or opted before in any of the research paper found during literature survey.

1.7 Chapter Summary

It is very important to cater the issues of BOTs in Software defined networks. The Paper identified issues and limitations found during literature survey and review in existing techniques. A problem statement has been identified and a new technique has been proposed to cater the limitations of existing techniques. The new technique and methodology proposed in this Paper will have a great positive impact in the research area for this specific identified problem area.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Chapter 2

Literature Review

A detailed literature survey has been performed having literature review of approx. 40 papers and approx. 10 of these are selected which are most recent, having impact factors and which are most related to the problem statement. Existing research has been systematically analyzed in breadth and depth. These papers are selected by the process of literature comprehension. Detail of selected papers is given below:-

2.1 State of the Art

A detailed working on the security of SDN Controllers has been done in [1] with a detailed introduction of Software Defined Networks, definition of controllers, Application programmable interfaces and recommendations to improve SDN Controller security. The concept of thesis is based on splitting the network intelligence out of the packet switching device and putting it into a logically centralized controller. The forwarding decisions are made by the controllers, which are located into the switches via standard protocols, like Open-flow. Thesis also discussed a comprehensive literature review concerning SDN controllers security has been done. Demonstrate a comprehensive studies on SDN by clarifying its concepts, Open-flow protocol architecture and how it works [1].

A discussion on SDN controllers and several threat Vectors which may enable for the exploitation of vulnerabilities of SDN Controllers. Designing a dependable Controller platform including the requirements for a secure, resilient, and robust SDN Controller. How we can secure SDN controllers by making recommendations for security improvements for future SDN Controller. Existing gap between the actual security level of the

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

current SDN Controller design and the potential security solutions. controllers securities are addressed by employing detection system to help in identifying any abnormal flows, trust model used with multiple trust anchor certification authorities and use of cryptography across controllers to secure the communication. Rule-chain Conflict Analysis (RCA) has been used for detecting the occurrence of any conflict [1].

Thesis concluded on the recommendations to improve the security of SDN Controllers with three design recommendations i.e. (1). based on software security principles (2) Secure default controller setting: Safe mode boot processes (3) Application Future Proofing [1].

Botnet classification using centralized collection of network flow counters in software defined networks has been proposed in [2] for the detection of botnets in software defined networks. The paper defined the bot-nets as a network of bot-malware infected machines controlled by bot-master is one of the top listed cyber security threat. Botnets are the most advance family of malwares and continuously changing to evade detection techniques. To cope up with changing nature of botnet, the detection techniques need continuous improvement and integration into new technologies. The latest techniques in botnet literature are targeting network header information only to classify botnet behavior using machine learning approaches. Paper also defined Centralized network flow collections as one of the major challenge in these techniques. The paper propose a botnet detection in SDN by collecting centralized network flow statistics in form of OpenFlow counters. The proposed approach apply C4.5 decision tree based supervised classification algorithm on collected counters. OpenFlow counters are suitable candidate to provide a good feature set to distinguished network behavior patterns of bot-malware. Paper also discussed the history of botnet, botnet impelled in early 90s from egg-drop application written to manage Internet relay chat (IRC) network. The botnet start with the same idea of centralized architecture and IRC protocol as used in eggdrop. These early botnets having centralized architecture disclose there c2c servers and are vulnerable to single point of failure. The centralized botnet introduce intermediate layers to hide c2c servers. The botnet change with time and introduce distributed and hybrid architecture. The work proposed in show evolving trend of botnets. The paper proposed detection model with Implement supervised decision tree based botnet detection technique into SDN Evaluate if OpenFlow counters provide enough statistics to detect botnet patterns. C4.5 supervised decision tree algorithm Used and train the model with

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

publically available botnet datasets. Approach consist of three stages .i.e (i) Centralized collection of OpenFlow counters from SDN controller (ii) Feature set extraction from collected counters (iii) Apply C4.5 supervised machine learning algorithm to classify flows into botnet flows or normal flows [2].

Work proposed in [3], apply three different supervised classification algorithms. The focus of the approach is to detect c2c servers therefore the technique first isolate server flows from incoming flows based on open listening ports. The server flows fed into all three detection algorithms for evaluation. The flow sized based, client access pattern based and temporal behavioral based features used in these detection algorithms. Another approach evaluate different flow intervals with proposed model and shows optimum results for flow monitoring interval. The work also include two botnet samples (Weladec and Blackenergy) in testing dataset to evaluate if their detection model is capable to cope up with unknown botnets. The evaluation results show that the model is capable to detect unknown botnets [3].

The work proposed by matija evaluate eight different supervised MLAs. The approach uses 39 different flow features for training and detection of bot malware patterns in provided dataset [3]. The proposed detection system in [4] evaluated with only two samples of Weladec and Strorm botnets. The approach shows very promising results. The work proposed in revisit network flow features used in machine learning based botnet detection techniques. The approach uses a dataset comprising of 16 different botnet samples. The dataset generated by combining three different real world datasets including ISOT dataset, ISCX 2012 IDS dataset, and botnet dataset of malware capture facility project. The approach evaluate different set of network flow features with this dataset. The botnet literature only have few proposed techniques that explore SDNs. These techniques collect flows from individual networking elements as applied in classical network and none of these techniques explores OpenFlow counters for botnet detection. The work proposed in COFFEE explore SDN for botnet detection. The approach work in two phases. The first phase collects netflows from individual forwarding elements and analyze to detect potential botnet flows. This approach proposed to use the same flow feature set as used in [4]. The second phase collect payload for the detected flows by using SDNs dynamic programmability. This phase classifies detected flows into botnet or normal. The work proposed in explore SDNs for botnet detection. The proposed approach used IPFIX flow protocol to collect network flows from each forwarding ele-

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

ments. The approach does not share evaluation results [4].

Paper in [5] presents the review of SDN security and potential threat against botnet Attack. It defines the BOTNET Definition, SDN Definition, CIA Definition, BOTNET Prevention in SDN, At the end of this paper, a future research to handle botnet attack. Paper also define the potential attack on SDN which includes Unauthorized Access, Data Leakage Access:- There is some possibilities attacker disguised as controller or application from a 3rd party. An attacker could get access to network resources and control the network operation and Data Leakage Packet handling: - SDN has several potential actions such as forwarding, drop, and send a packet to the controller. With this system, there is possibility attacker to determine the action to the specified packet by determining time process for packet arrived because packet forwarding from packet handling to the controller is longer than others. Other attack discussed are Data. Modification: - Attacker modify data whenever they can hijack controller. Attacker modify an inject flow rules in network devices, Malicious Application:- Because of SDN open for 3rd party application to the architecture, some application could be exploited by an attacker and drive into the unsafe state, Denial of Services, Configuration Issues:- SDN as the programmable network has weakness in security because it can come to be vulnerable especially for data or control communication and System Level SDN Security Its important for an operator to know the mode and switch activated in connection disruption, forwarding pattern during failures, the effect on flow entries and pattern of the controller when reestablishing the connection. Paper provided SDN Defense Mechanism against Botnets that includes generally, methods that have been done before on traditional network are based on the habit and pattern of network traffic. Paper discusses 5 main components with the following explanation: (i) Traffic Flow and Feature Extractor Module At this stage, the network traffic from different hosts is classified with the aim of obtaining information obtained from network flow, (ii) P2P Application Detection Module This module has the main purpose of obtaining the vector feature. In the process this module is divided into 4 sections as follows: (a) Building Detection Model and Training set (b) Feature Selection (c) Classifier (d) Traffic Analysis. (iii) Report to OpenFlow Controller. If the classification result of the detection agent matches the class on the P2P botnet, the detection agent will inform the rule arbitrator to adjust the flow entries in the data link bridges. (iv) Flow Rule Modify on OpenFlow Switch Once the Rule Arbitrator receives a RESTful HTTP request sent from the detection agent,

the flow table will be modified by a RESTful HTTP request in the data-link bridges. (v) Drop, Forward, or Redirect Packet [5].

Paper in [6] discussed the Machine learning based botnet detection in software defined networks. It used the flow based approach to detect botnet by applying machine learning algorithms to software defined networks without reading packet payload. Uses network flows as input and process it in two windows based modules to extract a statistical feature set to be used for classification. The first module process network flow stream to extract flow traces. The window size of this module is 10 which means a flow trace with 10 flows is considered as a trace of interest and forwarded to the next module for further processing. The second processes selected trace and fetches historical flows in last 60-minute window for the source and destination IPs of the trace. feature set is extracted from selected trace and relevant historical flows. The approach applies supervised decision tree based machine learning algorithm to create a model during a training phase using extracted feature set. This model is then used to classify flow traces during the testing phase. dataset for experimentation is extracted from publicly available real botnet and normal traces. The experimental findings show that the method is capable to detect unknown botnet. The results show detection rate of 97 percent for known botnets and 90 for unknown. The paper proposes to use intelligence from both real-time flow stream and historical context of a flow trace to extract a more meaningful feature set to help classification process. C4.5 a decision tree based supervised machine learning algorithm for botnet classification. The proposed approach operates in two modes the training mode and test. This is used to train the C4.5 algorithm to recognize botnet behavior patterns with help of labeled dataset. A decision tree model is created based on training during this mode. The testing uses pre-computed model to classify flow traces. The botnet are collected from publicly available CTU-13 dataset and ISOT. For normal traffic representation, the flow traces of ISOT dataset is used for training and testing purposes. Flow trace detection module is responsible to assemble a batch of flows two network endpoints. A flow trace is an ordered sequence of flows between two network endpoints. This module uses key $<$ Source IP, DST Internet address, DST port, Protocol $>$ to identify flows of two endpoints for the same application. flows with same key are grouped together to form a batch. Each flow of a batch is counted and when a batch reaches to the count of 10 flows that batch is exported to the feature extraction module. Any subsequent flow of the exported batch is recognized as a new trace and go

through the batch assembly process. The count of 10 flows is selected on bases of flow interval research in the literature which varies between a range of 10 to 60 flows. From trace source address or pointed to trace destination addresses in last 60-minute duration. The source and DST ADDR of the trace of interest are provided by feature extraction module. These addresses used as matching criteria to fetch flows from the controller. These flows are then forwarded to feature extraction module for further processing [6]. It Works with two time based separated batch of flows. The goal of this module is to extract statistical features from current network activity and historical context of the same. To achieve this two batch of flows are collected, first batch of 10 flows from trace detection module and second batch is collected from historical flow collection.

classification module uses C4.5 decision tree based supervised algorithm. The input to module is global feature vector generated by extraction It operates in two different modes. The system initially run in the training mode and the global feature vectors with pr-assign labeled to each fed into the classification module. C4.5 algorithm learn with help of provided labeled and create a decision tree model. This is then used in testing mode to detect botnet. Each input of feature vector during testing mode traverse through decision tree model and get assign a label. Real network traces for botnet and normal traffic are used to perform experiments. Six pcap files of different botnet families including Neris, Rbot, Virut, Menti, Murlou, and Sougo are collected and one pcap files for normal traffic [6]. Phase-1 for the training mode and phase-2 for the testing mode. The traffic for testing purposes has 50 percnet representation of unknown bot families for which the decision tree model is not trained. This is to test if proposed model can detect unknown botnet and to reduce dataset biasness. Experimental setup prepared using SDN controller opendaylight, the time series data repository (tsdr), the tsdr feature of opendaylight makes flow state collection abstract for user applications. The custom application only requires communicating with tsdr for collection of either flow stream or historical time series of flow stats. The prototype of the proposed system is developed in java. To simulate the dataset for evaluation, a network emulator tool called mininet is used to create Virtualized network infrastructure. This infrastructure is then used to simulate the dataset for training and testing purposes. Experiments are performed using labelled ground truth data and testing data. The results are compiled in two steps. The first step concludes the result at individual trace of interest granularity. This help to analyze system performance for individual botnet family sample included in testing

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

dataset. Three known botnet family samples and three unknown botnet family samples. The botnet samples including Rbot, Virut, and Menti are considered known botnet as detection model is trained for these whereas Neris, Murlo, and Soguo are considered as unknown botnet. The average detection rate of the proposed method for known botnet is 97.1 percent and for unknown botnet is 90.4 percent. The second step conclude results to analyze overall system performance. Overall accuracy of the proposed method is 94.8 percent. The paper concluded the proposed work detect botnet in software defined network in near real-time. The delay is approximately 10 consecutive flows counted from the first flow of a botnet trace. To increase detection rate and reduce false positives, the method uses a rich feature set extracted from current network activity of a trace and historical context of the same. The works shows promising results with detection rate of 97 percent for known and 90 percent for unknown botnet. The detection rate is much higher as compare to other approaches with same level of diversity in testing data. The proposed system shows accuracy of 94.8 percent. The system uses TSDR feature of opendaylight to support the concept of stats plan in SDNs. Separating statistics from control plan not only reduce computation load on controller but also provide a centralized statistics visibility. In context with stats plan, the future extension of this work is to reflects its computational results back to the stats plan so that other application may use this information [6].

It is important to discuss the performance of Bot-net detection by Neural Networks in SDN Same is done in paper [7]. It use Neural Networks to detect Bot-net in SDN. ANN classifier trained by available data sets collected in conventional networks. Accuracy of Bot-net detection higher than 99 percent.NN using the Kohonen algorithm trained to recognize unauthorized activities in a SDN. Managed from Open-daylight (ODL) controller in combination with Network Function Visualization (NFV) has been also proposed. It is based on a Self-Organized Map (SOM) learning method for the classification problem phase to build an IPS for DDoS attacks mitigation. The fundamental problem in NN-based botnet detection is the definition of suitable discriminators to detect malicious traffic against regular data in the network. Two hundred discriminators as the set of most Significant ones. A botnet classification strategy based on the centralized collection of network flow counters in SDN, and the use of a supervised C4.5 decision tree classification algorithm, is proposed. Analyzing only the Open-flow 1.3 messages exchanged. Traffic classification by an artificial neural network (ANN) to

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

identify Bot-nets in a SDN as playground. Such ANN is trained with a data training set (TS) obtained by a Bot-net attack running in conventional networks (CN), i.e. non SDN. Extracts a set of relevant parameters, referred to as discriminators. Supervised machine learning problem by using a Training Set (TS) and a Test Set with features computed from real traffic data with known malicious/non-malicious label. As features are numerical, we thought it was appropriate to use a Multilayer Perception (MLP), which is a classical and easily manageable ANN architecture, trainable with the error back-propagation weight update rule. Paper aimed to identify a small MLP that, after training, can be used as a real-time detection engine for Bot-nets. Determine the minimum number of hidden neurons in a MLP with a single layer that achieved an acceptable error rate. This preliminary study is particularly important for the Bot-net detection task. As a large number of neurons in the MLP results in longer computation time that may negatively affect the real-time performance of the desired detection engine. Paper found out that just 5 hidden sigmoid neurons were sufficient to achieve an acceptable error with a small improvement by using a higher number of hidden neurons. Consequently tried to increase the layers, just to discover a small improvement in the performance of Bot-net detection. kbDetector interactions with the SDN controller can be sketched as follows:- (i) The switch sends the Open-flow packet ofp in to SDN controller. (ii) replies to switch dictating to create a Flow Entry, and notifying, via Web-socket to kbDetector, that a new flow has started (iii) when hard-timeout expires, the switch sends a ofp flow removed Open-flow message with the statistics to the controller which then sends a notification via Web-socket to kbDetector that contains the statistics and the stream id. (iv) kbDetector calculates the features from incoming information and activates the trained MLP (v) If the MLP classifies flow as malicious, kbDetector retrieves the MAC address of the internal hosts involved in the flow and via REST messages add a block rule of flow in order to isolate the infected host in the appropriate SDN network switches flow entry. Developed the application kbTool, manages host blocked. Communicates via REST API with the controller in order to share configuration lies with kbDetector. Experiments for some selected MLP architectures that are reported for both CNs and SDN. All the MLP architectures were tested with different sizes of the time-window. main result we obtained is that an MLP trained over data concerning non-SDN has a very satisfactory performance also when tested for Bot-net detection in SDN. As expected, though, the test error in SDN is higher than that for non-SDN

they were trained for. Longer time windows bring about a better test error for most architectures, and that a four layers MLP is good enough to achieve an error rate of less than about 0.05percent. For experiments, paper compute features based on portions of packet trace until the layer 4 (ISO/OSIstack), thus the flows have same characteristics in SDN and CN, are completely transparent to an observer. The difference is in the specific set of discriminators used to train the ANN in two cases of CN and SDN. The MLP in experimentation were trained over a TS obtained by joining public datasets that are commonly used in the literature for Bot-net malicious traffic identification. The rest is CUT-13 dataset that come from different categories of Bot-nets. Then use the ISOT dataset [12] by taking out a small percentage of malicious traffic. Data-set have large number of flows in pcap (packet capture) format. Subdivision into small pcap files via Split-cap 3 for division of flows. ISOT dataset 9.9 GB contains 914812 different flows. Time windows: 10, 30, 60, 120, 180, 240, and 300 seconds. For each time window, 70percent of the files were used for the creation of the training set and the remaining 30 percent for the test set. New scenarios from the Stratosphere IPS 4 were added to the test set. training and test files have been splintered by duplicates. MLP was stopped either when the error was considered acceptable (0 percent) or when a maximum number of 1500 training epochs was reached. Performance of the trained MLP is measured by means of the confusion matrix with its four quadrants that represent TP, FP, TN and FN, Actual positives result P = TP+FN, Negatives N = TN+FP. Confusion matrix metrics resulting from are presented for the selected ANN architecture. The performance metrics for a supervised NN are computed from the four quadrants of the confusion matrix: the model accuracy is defined as A=(TP+TN)/(P+N), Precision as S=TP/(True positive +FP), and the recall as Re=TP/P= TP / (TP + FN). The values of these performance metrics obtained in tests are shown vs. the time window's duration for both CN and SDN with protocol OpenFlow 1.3, for the selected ANN architecture. The experience of last October, when Mirai Botnet DDoS 17 Dyn Data Centers (DCs), repair sent a case study. Most dangerous threat for the performance of a DC is represented by a DDoS attack. Many cyber criminals started using the tool to assemble their own botnet armies. Paper simulated the same event in a SDN environment, by implementing a typical Fat-tree topology to create a Software-Defined Data Center (SDDC) with 4 pods within the Mininet framework. Simulator have infected with the Mirai malware six hosts in pods 1, 2 and 3, by listening on TCP ports 23 and

2323, in order to simulate vulnerable IoT web cams. One of the infected hosts plays the role of Mirai CnC server with a MySQL DBMS support, for scan receiver function used by Mirai. The victim host is located in pod 0, and undergoes two different DOS attacks: SYN Flood and UDP. Experimentation relies on the evaluation of the traffic flows in the network order to generate the set of relevant features to be given in input to the ANN for Bot-net detection Evaluate the performance of approach in terms of confusion matrix metrics accuracy, recall and precision for Mirai detection. A sample of traffic between CnC and bots detected in a time window of 180 seconds. In particular, the packet set (in both directions) is represented by a quintuple consisting of source address, destination ADDR, SRC and DST PORT and protocol at the transport level. Paper analyze the efficiency of trained ANN as a Bot-net detection tool by simulating UDP flood attacks by bots replicated from the Mirai malware. Obtained the best performance in the attack detection for time windows of large duration, as expected, and for time windows of 240 or 300 seconds the performance is roughly the same. In practical implementations, the only possible time window with the current version 1.3 of OpenFlow protocol is 300 seconds. However, a real-time system for botnet detection would benefit significantly from the use of time-window sizes few seconds, Especially if measures, like, e.g., isolation from the rest of the network, should be taken. The main reason is that the current protocol is not provided with a method for extracting flow statistics at arbitrary time intervals, and the only way is by removing a flow entry from the switch. Furthermore, in a high traffic environment, such as DCs, that manage a large number of streams, it is mandatory to reduce of rule removals per second. So, there is no choice but setting the time window at the highest possible duration. Paper concluded by Statistical analysis and classification by a supervised neural network is an effective method for detecting the malicious traffic produced from bots during the attacks, for individuating the communication flow between bots and CnC for preventing the attacks. In addition, it have shown that it is possible to block the attacks at the source, not simply a mitigation strategy. During the testing phase we noticed, like a serendipity, that DDoS attacks were automatically recognized by neural network despite the work is explicitly focused on Bot-net detection and not on the application. In particular, the use of neural networks has been as effective as other Machine Learning methods with 99 percent accuracy. The use of OpenFlow protocol version 1.3 imposes some limits, like e.g., the impossibility of performing real-time detection, due to the constrained time

window duration. Future Work:- Open-flow 1.5.1 will be used, analysis of performance of the detection application by implementing the kbDetector with Big-Data techniques and the implementation of a malicious traffic generator to improve the results [7].

It is very important to get the knowledge of BOTNET definition, all BOTNETs classes that exists and can affect software defined networks. Paper discussed in [8] describe all type of BONET classifications available. Paper discussed that only few formal studies have examined the botnet problem and botnet research is still in its infancy. In this survey botnet detection techniques based on passive network traffic monitoring are classified into four classes including Signature-based Anomaly-based DNSbased Mining-base Signature-based techniques can only detect known botnets, whereas the other classes are able to detect unknown bots. However, most of the current botnet detection techniques work only on specific botnet CnC communication protocols and structures. According to the comparison, the most recent botnet detection techniques based on data mining as well as DNSbased botnet detection approach can detect real-world botnets regardless of botnet protocol and structure with a very low false positive rate. Hence, developing techniques based on data mining and DNS traffic for botnet CnC traffic detection has been the most promising approach to combat botnet threat against online ecosystems and computer assets [8].

Work done in [9] uses POX Controller, IPFIX template to detect bots, Offline analysis, Machine Learning approach, Categorization with Source IP, Destination IP, Ports, protocols, interface anc class of service, number of packets, number of bytes, MLA used to detect bots, Flow collector, Multistage filtering to remove unnecessary data in five steps (i) detect IRC botnets (ii) Remove port scanning data (iii) remove high bit rate data flows (iv) remove flows with two or less packets, Botnet Detection Engine:- No MLA mentioned in the paper [9].

A research article [10] included in this document to get the idea of detecting point-2-point Botnet in Software defined networks. The solution use Machine Learning Approach. The captured packets with the same 5-tuple (i.e., source IP address, source port number, destination IP address, destination port number, and protocol) information and packets with reverse direction will be recognized as the same flow if they occur closely within a short time period. After a flow has been classified, the Detection Agent reports the result to the Rule Arbitrator with the 5-tuple information and the type of P2P botnet or application. The Rule Arbitrator, afterwards, modifies the related flowta-

bles in the Data-link Bridges in accordance with the result reported by the Detection Agent. Finally, the Data-link Bridges automatically drop the malicious packets that are recognized by the classifiers. It use Netmate, an open source tool, to capture packets and transform them into traffic flow, from which we extract feature vectors for machine learning analysis. This tool has been frequently used to capture network packets. Need to define the time frame to capture packets (i.e., a flow duration). Test different flow durations and analyze the performances. Based on the result, it found that the flow duration with 600 seconds has the best performance. Article propose a system which can detect and categorize P2P traffic in SDN with machine learning, automatically and flexibly adjust flow entries to manage network traffic, and thus reduce the load of network administrator. Experiment system in a test bed to evaluate the performance of classification accuracy and traffic management. Experiment results show that our system can detect all the considered types of P2P network traffic with high accuracy rate and automatically manage traffic with flow entries through SDN controller [10].

2.2 Limitation of Existing Techniques

The solution in this Paper is proposed after having a detailed literature survey to make sure that proposed technique covers all the limitations of existing technique with best possible method.

There are several limitations in technique proposed in [2], i.e. (i) Limited to OpenFlow protocol only (ii) Limited to only Open Daylight controller (iii) Only for known botnets 80 percent results accuracy (iv) Limited Publically available datasets Used (v) Only C4.5 Decision Tree used (vi) Feature extraction use feature set SP, DP, Protocol, Duration, Bytes, Packets, BPS, PPS, BPP, Pit (vii) Only detection is addressed not prevention (viii) Slow due to extensive computations for each flow (ix) Limited to specific types of botnets. Paper discussed in [5] is a survey paper, not reached to a single conclusion, limited to specific types of botnets and proposed only detection not prevention.

There are several limitations identified in paper given in [6] i.e. (i) A resource hungry process proposed as each time software have to identify 10 unique flows to form a single label, then search for 60minute traffic for this label. Then compare this label with last 60 minute traffic. (ii) Each time same process need to adapt and no record of successful detection given as feedback to the system for blocking without any further computation.

(iii) Only tested for open day light controller (iv) Only tested for OpenFlow (v) Only detection is addressed not prevention (vi) Limited to specific types of botnets. Solution proposed in Paper [9] is a resource hungry process as high computations involved, Multistage filtering again involve processing delay, Not real time, doing offline analysis, Only tested for POX controller, Only tested for IPFIX Templates, Only detection is addressed not prevention, Limited to specific types of botnets.
Paper in [7] has solution with resource hungry process proposed, each time same process need to adapt and no record of successful detection given as feedback to the system for blocking without any further computation. Only tested for specific controller. Only tested for OpenFlow 1.3. Analyzing only OpenFlow 1.3 messages exchanged. Use of non sdn networks for training data. What if new bot come in network? High time window required to get results. Higher error rate of proposed solution for SDN as compare to CN.

2.3 Chapter Summary

Literature survey is the most important part of any research to make sure that the proposed work is not done before and can cover the limitations of existing techniques. This chapter provides the detail of literature survey done for this Paper to make sure that proposed methodology caters the limitations and issues of existing techniques. First part of this chapter provides the detailed literature survey and then explained the limitations of existing techniques.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Chapter 3

Proposed Solution

This Paper proposed a DNS based approach to detect and prevent botnet by applying one million web database to software defined networks without reading packet payload. The proposed solution maintain good one million website database (1Mdb). To do any activity, Bot need to communicate with CnC and requires DNS to IP resolution. For any request having destination port 53 (DNS) will be checked. The protocol will get all matching traffic and send to 1Mdb. If URL Exists in 1Mdb then do not respond otherwise send reply with remove flow and block flow to the controller. The approach have a lot of benefits as compare to other approaches i.e. Very less computations on live traffic, Only Traffic for Port 53 will be monitored, No computation or processing on remaining traffic, Bots will be blocked at very start of communication, Infected IPs will be permanently blocked, System not only provide detection but also prevention solution, standard Technique for all type of controllers and protocols, isolate infected system before start of communication (Proposed solution architecture is shown in Figure-3.1 Below).

3.1 Proposed Solution Architecture Components

This section explains the architecture components that are used in the proposed solution and methodology.

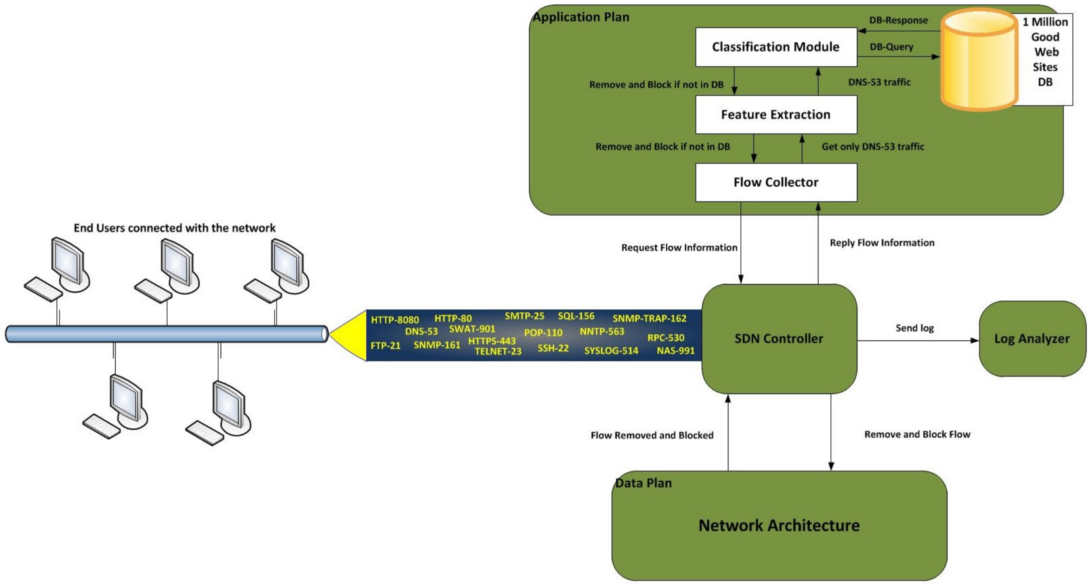


Figure 3.1: Proposed Solution Architecture Components

3.1.1 End User

End user is the part of solution architecture which is generating the user traffic and for which Bot detection will be processed.

3.1.2 SDN Controller

Controller is the heart of SDN network which control each and every packet to/from SDN networks. In the propose solution, the open day light controller has been used. The controller is used to extract the required feature set by flow request and to block/remove flow as per decision done by application plan.

3.1.3 Flow Collector

Feature collector module is one of the most important module of the proposed solution. This module have to collect each and every packet of the network with all header information. The module will cover all protocols to collect form each part of the Software Defined Network via Controller to make sure that all traffic passed to the proposed solution for further processing. As this module have to process huge amount of data so it must be very accurate and fast to avoid any packet loss. After processing at its end, this module will forward traffic to feature extractor module for further processing.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

3.1.4 Feature Extractor

Feature extractor module will collect the selected features from all the flows coming from flow collector. In our case only dns traffic on port 53 will be extracted. The source IP address, destination protocol, destination service and destination URL will be extracted as feature set.

3.1.5 Classification Module

The classification module will classify either the URL is a legitimate website or related to CnC botnet traffic. For this the classification module query from the one million good website database. If the answer found yes then it will classify it as legitimate and no action will be taken otherwise the flow will be considered as bot-traffic and will be blocked and removed. Naive Bayes Classifier can be used to classify the data.

3.1.6 One Million Good Websites database

A one million database has been used in the proposed solution to cross check each dns request from one million good website database. If the requested url exists in this database then no further action need to be taken otherwise respective flow will be removed and blocked for further communication. This database is available to download from http://s3-us-west-1.amazonaws.com/umbrella-static/index.html and is regularly updated.

3.1.7 Log Analyzer

Log analyzer has been added in the proposed solution to keep log of each flow that has been removed or blocked from the network to keep track of traffic that is not passing from the network.

3.1.8 Dataset

The selection of dataset is very important task for machine learning based botnet detection and prevention techniques. The poor selection of dataset possibly lead to biased results. The real world and publically available dataset is a good choice for evaluation

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

of botnet detection techniques. To meet these criteria, we will use publicly available CTU-43 botnet dataset. This dataset provide packet dumps (pcap files) of seven real botnets (Neris, Rbot, Virut, Murlo, Menti, Sogou, and NSIS). We will use these files to generate botnet traffic for evaluation and test our model. To generate normal traffic, we selected ISOT dataset. This dataset provides a single pcap file having normal traffic and traffic for weladec and zeus botnet. Other dataset includes the one million website datasets as well. Brief detail of these datasets are given below:-

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Cisco_db:19:c3	Broadcast	ARP	60	Who has 147.32.84.165? Tell 147.32.84.1
2	8.982709	Cisco_db:19:c3	Broadcast	ARP	60	Who has 147.32.84.165? Tell 147.32.84.1
3	50.099564	Cisco_db:19:c3	Broadcast	ARP	60	Who has 147.32.84.165? Tell 147.32.84.1
4	50.369266	54:52:00:00:00:01	Broadcast	ARP	60	Who has 147.32.84.165? Tell 147.32.84.85
5	51.369054	54:52:00:00:00:01	Broadcast	ARP	60	Who has 147.32.84.165? Tell 147.32.84.85
6	52.369688	54:52:00:00:00:01	Broadcast	ARP	60	Who has 147.32.84.165? Tell 147.32.84.85
7	53.086840	Cisco_db:19:c3	Broadcast	ARP	60	Who has 147.32.84.165? Tell 147.32.84.1
8	59.086131	Cisco_db:19:c3	Broadcast	ARP	60	Who has 147.32.84.165? Tell 147.32.84.1
9	160.084662	PcsCompu_b5:b7:19	Broadcast	ARP	60	Gratuitous ARP for 147.32.84.165 (Request)
10	160.084668	PcsCompu_b5:b7:19	Broadcast	ARP	60	Gratuitous ARP for 147.32.84.165 (Request)
11	161.077511	PcsCompu_b5:b7:19	Broadcast	ARP	60	Gratuitous ARP for 147.32.84.165 (Request)
12	161.077519	PcsCompu_b5:b7:19	Broadcast	ARP	60	Gratuitous ARP for 147.32.84.165 (Request)
13	162.079007	PcsCompu_b5:b7:19	Broadcast	ARP	60	Gratuitous ARP for 147.32.84.165 (Request)
14	162.079013	PcsCompu_b5:b7:19	Broadcast	ARP	60	Gratuitous ARP for 147.32.84.165 (Request)
15	162.765245	147.32.84.165	147.32.84.255	NBNS	110	Registration NB SARUMAN<00>
16	162.765253	147.32.84.165	147.32.84.255	NBNS	110	Registration NB SARUMAN<00>
17	163.510681	147.32.84.165	147.32.84.255	NBNS	110	Registration NB SARUMAN<00>
18	163.510692	147.32.84.165	147.32.84.255	NBNS	110	Registration NB SARUMAN<00>
19	163.926344	PcsCompu_b5:b7:19	Broadcast	ARP	60	Who has 147.32.84.1? Tell 147.32.84.165
20	163.926354	PcsCompu_b5:b7:19	Broadcast	ARP	60	Who has 147.32.84.1? Tell 147.32.84.165
21	163.929830	Cisco_db:19:c3	PcsCompu_b5:b7:19	ARP	60	147.32.84.1 is at 00:1e:49:db:19:c3

Figure 3.2: CTU-43 Dataset

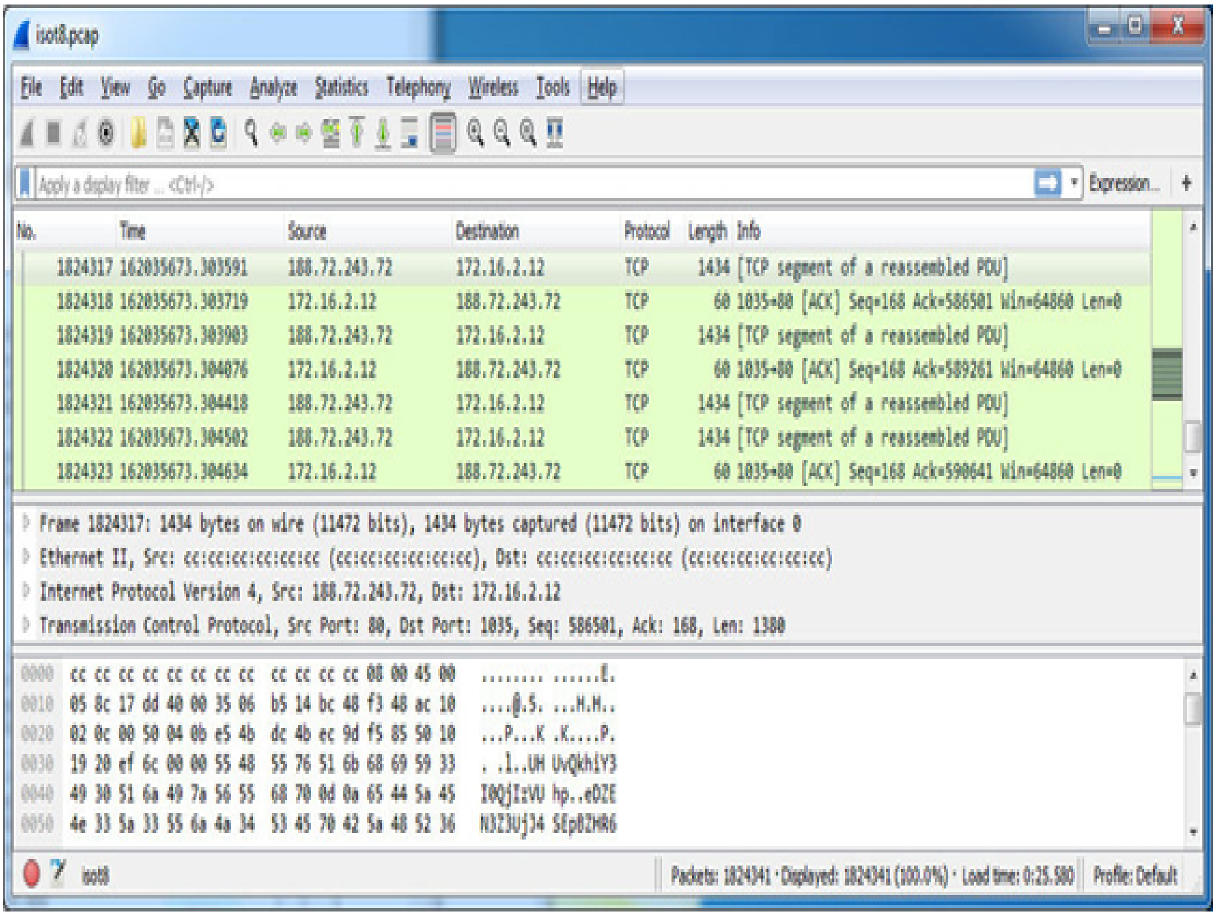


Figure 3.3: ISOT Dataset

SN	Website
1	netflix.com
2	api-global.netflix.com
3	prod.netflix.com
4	push.prod.netflix.com
5	google.com
6	www.google.com
7	microsoft.com
8	doubleclick.net
9	g.doubleclick.net
10	safebrowsing.googleapis.com
11	facebook.com
12	ichnaea.netflix.com
13	googleads.g.doubleclick.net
14	google-analytics.com
15	clients4.google.com
16	data.microsoft.com
17	live.com
18	apple.com
19	clientservices.googleapis.com

Figure 3.4: One Million DB Dataset

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

3.2 Comparison

A brief comparison of existing solutions and proposed one is given below:-

Reference	Header + Data	Analyze each packet	Complex	Performance	Approach	Standardized for all Controllers	Algorithm	Protocols Analyzed	Blocking of BOTs	Protection
DISCLOSURE[2]	Header only	Yes	Yes	Slow	Machine Learning	No	C 4.5	All	During or after communication	Detection Only
DISCLOSURE[5]	Header + Body	Yes	Yes	Slow	Honeypot	No	Honey pot	All	During or after communication	Detection Only
DISCLOSURE[6]	Header only	yes	Yes	Slow	Machine Learning	No	C 4.5	All	During or after communication	Detection Only
DISCLOSURE[7]		Yes	Yes	Slow	Neural Net classier	No	Kohonen algorithm C4.5	All	During or after communication	Detection Only
Proposed Solution	Header Only	No	No	Fast	Machine Learning	Yes	Naive Bayes Classifica[illegible]	DNS only	Before Start of any communication	Detection and Prevention

Figure 3.5: Comparison

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Chapter 4

Methodology, Implementation and Results

4.1 Deployment and Implementation

4.1.1 Flow Chart

Flow chart is given in this sub-section to explain the exact process flow for proposed solution. Same flow has been adopted to implement the system. User data will first reach to controller via control plan of Software Defined network. The data then will be collected by flow collector and will send to packet filter only for DNS traffic on destination port 53. All DNS traffic will be matched with 1Million good web sites database. If destination URL found in 1Mdb then no action will be taken otherwise flow will be forwarded to data plan for further necessary action to block and remove the flow and all its associated traffic. Flow chart is given below.

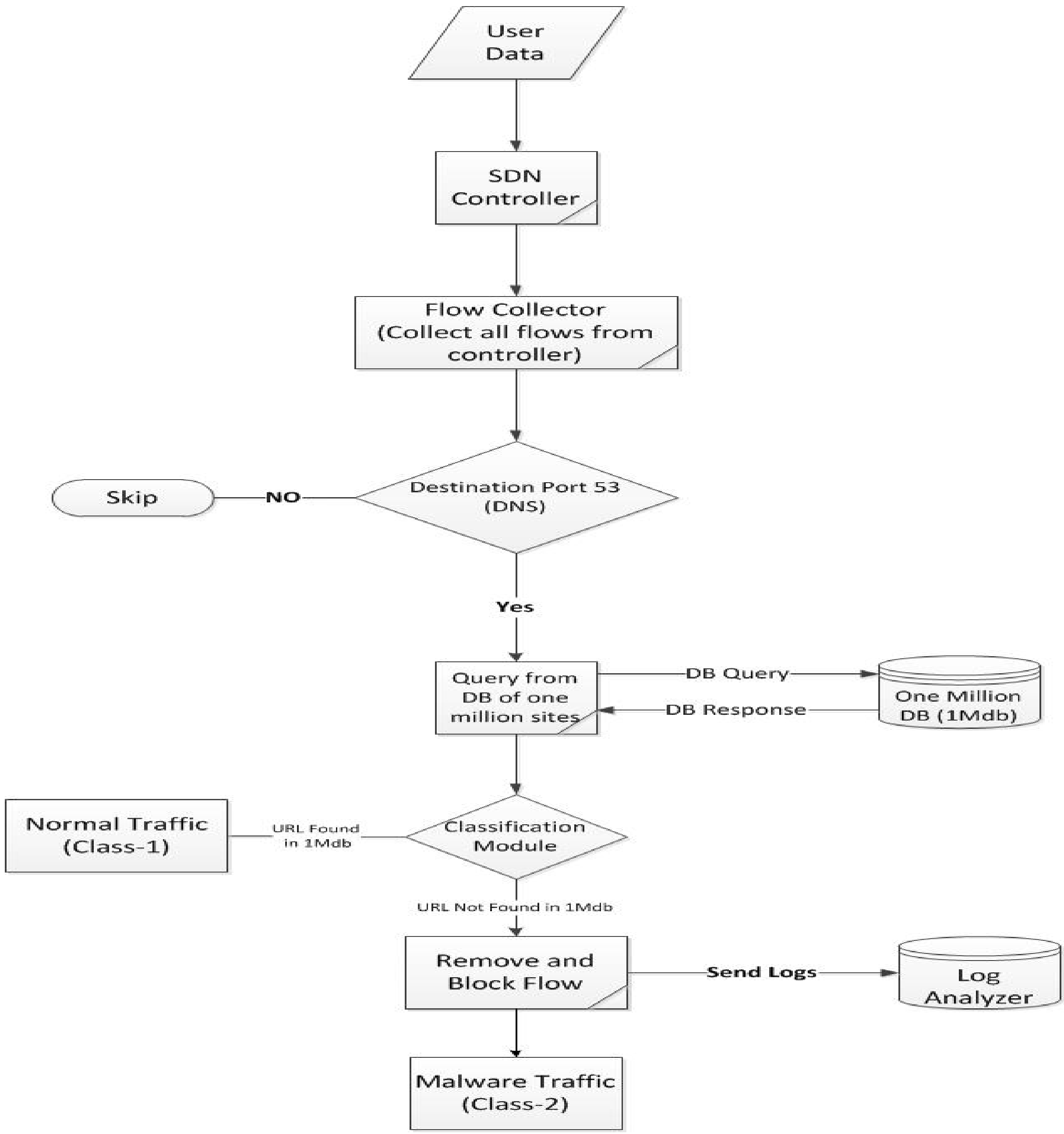


Figure 4.1: Flow Chart

4.1.2 Implementation

The implementation of the proposed solution will be performed by using python programming language. All the datasets will be imported to verify the traffic and to take appropriate decisions on the results.

4.1.3 Algorithm

Before implementation the program in the python programming language, an algorithm has been written so that same steps can be reflected in the code.

1. Import required libraries
2. Import 1Mdb datasets
3. Import Malware datasets
4. Import normal traffic datasets
5. Apply filter on destination port 53
6. If destination port = 53 then query the destination URL from 1Mdb else no action
7. If URL found in 1Mdb then no action required and mark traffic as Normal Traffic (Class-1)
8. If URL not found in 1Mdb then remove and block flow from data plan, send log to log analyzer and Mark traffic as Malware Traffic (Class-2).

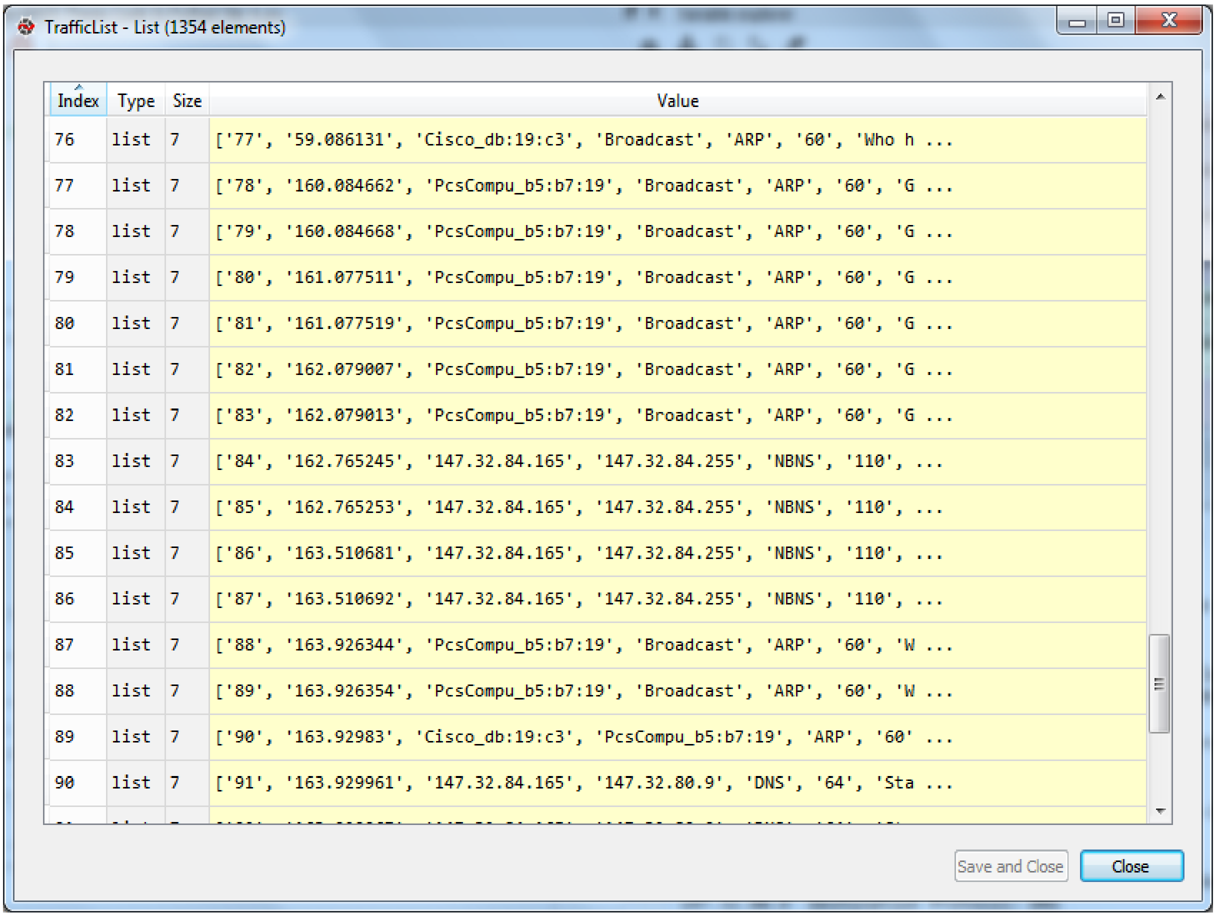
https://sites.google.com/site/ijcsis/
ISSN 1947-5500

4.2 Results

This section provide the results that are computed via program written in python language. Output of each step has been given in each subsection below then concluded on the final output of the program with the classification of user traffic.

4.2.1 Traffic List

All traffic data has been imported by using datasets mentioned in previous sections. Below is the sample import snap of dataset taken from the output of python program:-



TrafficList - List (1354 elements)

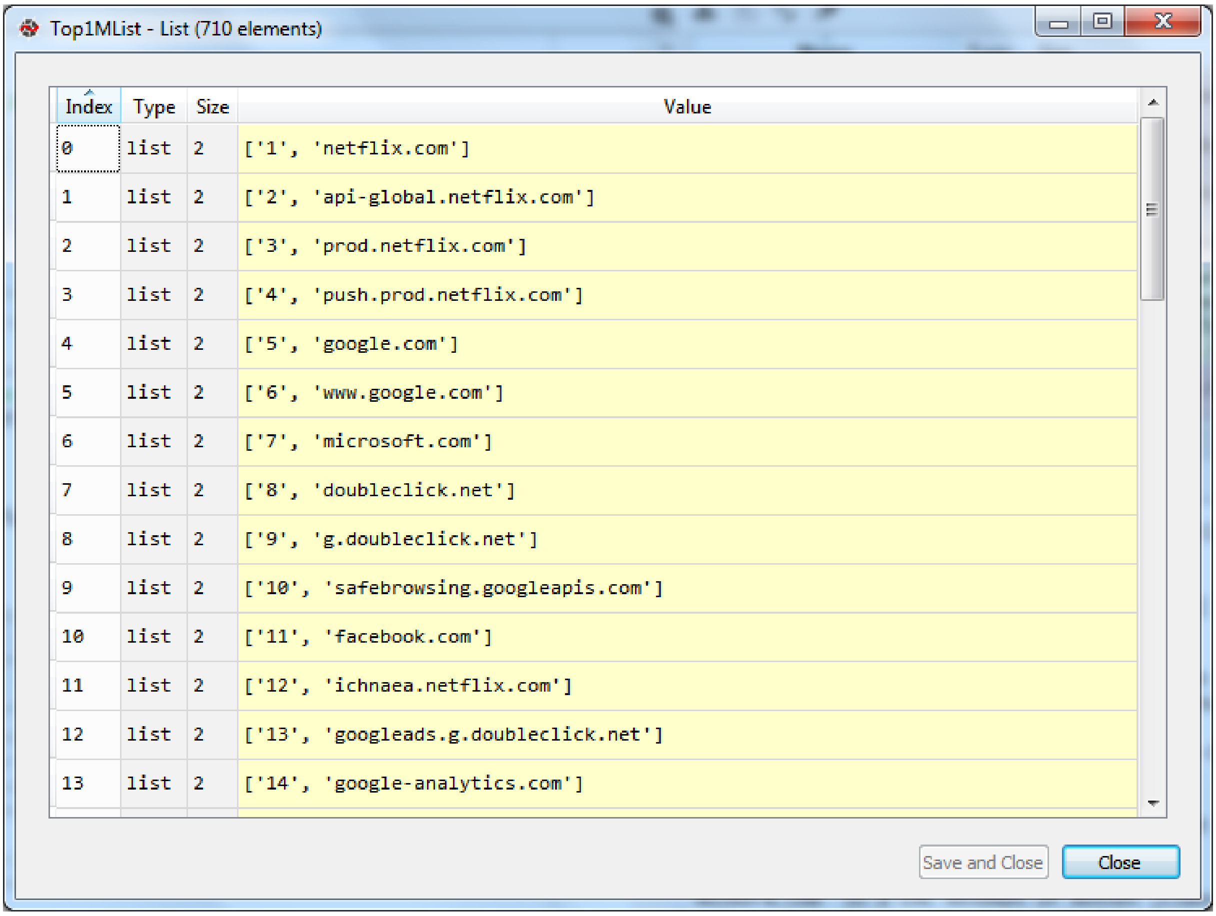
Index	Type	Size	Value
76	list	7	['77', '59.086131', 'Cisco_db:19:c3', 'Broadcast', 'ARP', '60', 'Who h ...
77	list	7	['78', '160.084662', 'PcsCompu_b5:b7:19', 'Broadcast', 'ARP', '60', 'G ...
78	list	7	['79', '160.084668', 'PcsCompu_b5:b7:19', 'Broadcast', 'ARP', '60', 'G ...
79	list	7	['80', '161.077511', 'PcsCompu_b5:b7:19', 'Broadcast', 'ARP', '60', 'G ...
80	list	7	['81', '161.077519', 'PcsCompu_b5:b7:19', 'Broadcast', 'ARP', '60', 'G ...
81	list	7	['82', '162.079007', 'PcsCompu_b5:b7:19', 'Broadcast', 'ARP', '60', 'G ...
82	list	7	['83', '162.079013', 'PcsCompu_b5:b7:19', 'Broadcast', 'ARP', '60', 'G ...
83	list	7	['84', '162.765245', '147.32.84.165', '147.32.84.255', 'NBNS', '110', ...
84	list	7	['85', '162.765253', '147.32.84.165', '147.32.84.255', 'NBNS', '110', ...
85	list	7	['86', '163.510681', '147.32.84.165', '147.32.84.255', 'NBNS', '110', ...
86	list	7	['87', '163.510692', '147.32.84.165', '147.32.84.255', 'NBNS', '110', ...
87	list	7	['88', '163.926344', 'PcsCompu_b5:b7:19', 'Broadcast', 'ARP', '60', 'W ...
88	list	7	['89', '163.926354', 'PcsCompu_b5:b7:19', 'Broadcast', 'ARP', '60', 'W ...
89	list	7	['90', '163.92983', 'Cisco_db:19:c3', 'PcsCompu_b5:b7:19', 'ARP', '60' ...
90	list	7	['91', '163.929961', '147.32.84.165', '147.32.80.9', 'DNS', '64', 'Sta ...

Save and Close | Close

Figure 4.2: Actual User Traffic

4.2.2 Top One Million Web Site Database

Top one million web site database has been imported by using the authentic data as mentioned in the dataset section of this document. Below is the sample import snap of dataset taken from the output of python program:-


Top1MList - List (710 elements)

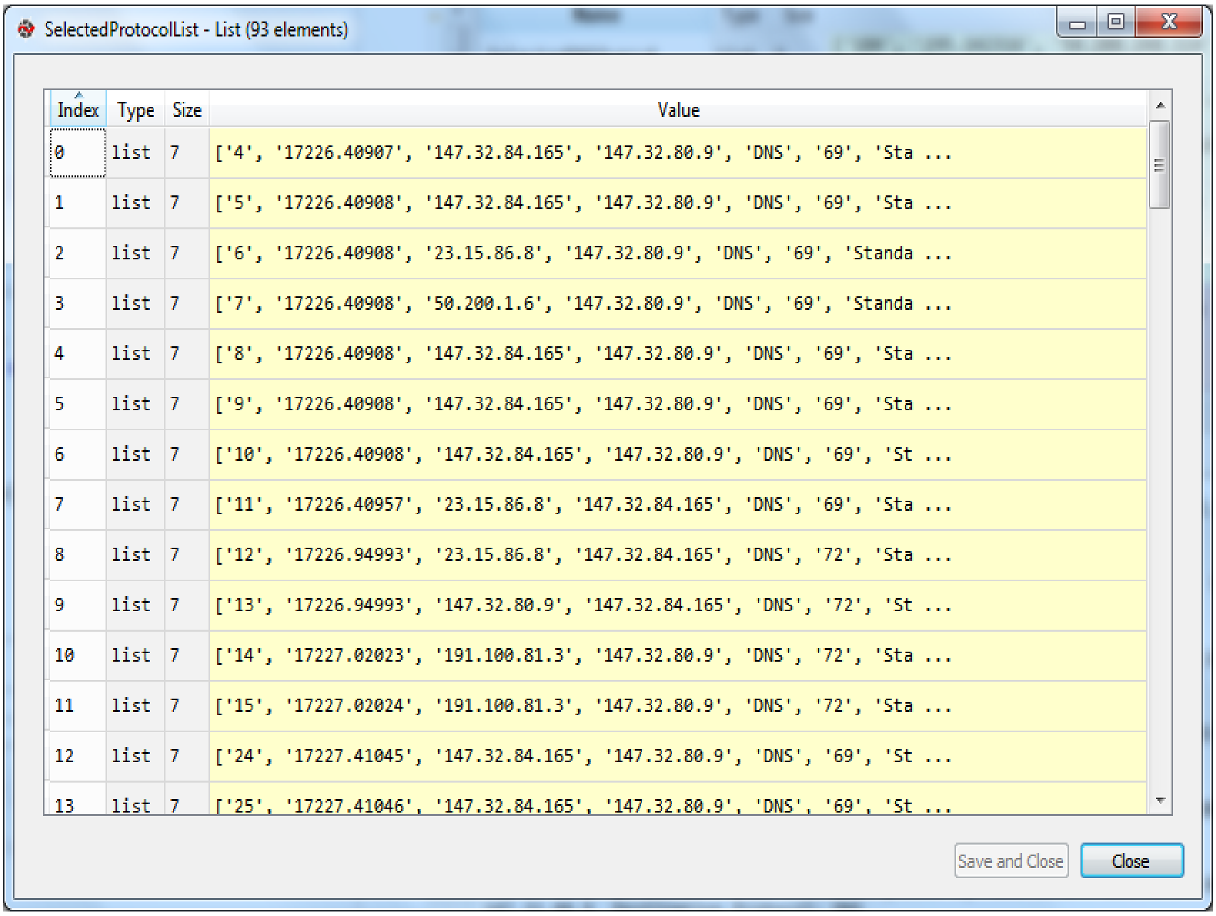
Index	Type	Size	Value
0	list	2	['1', 'netflix.com']
1	list	2	['2', 'api-global.netflix.com']
2	list	2	['3', 'prod.netflix.com']
3	list	2	['4', 'push.prod.netflix.com']
4	list	2	['5', 'google.com']
5	list	2	['6', 'www.google.com']
6	list	2	['7', 'microsoft.com']
7	list	2	['8', 'doubleclick.net']
8	list	2	['9', 'g.doubleclick.net']
9	list	2	['10', 'safebrowsing.googleapis.com']
10	list	2	['11', 'facebook.com']
11	list	2	['12', 'ichnaea.netflix.com']
12	list	2	['13', 'googleads.g.doubleclick.net']
13	list	2	['14', 'google-analytics.com']

Save and Close Close

Figure 4.3: Top One Million Database

4.2.3 Selected Protocol List

User traffic then filtered by using feature extraction module. As the algorithm is based on the filtration of user traffic on DNS protocol. So after filtration, only DNS traffic will remain to be processed for next step. Below is the sample output of feature extraction step that is taken from the output of python program:-



SelectedProtocolList - List (93 elements)

Index	Type	Size	Value
0	list	7	['4', '17226.40907', '147.32.84.165', '147.32.80.9', 'DNS', '69', 'Sta ...
1	list	7	['5', '17226.40908', '147.32.84.165', '147.32.80.9', 'DNS', '69', 'Sta ...
2	list	7	['6', '17226.40908', '23.15.86.8', '147.32.80.9', 'DNS', '69', 'Standa ...
3	list	7	['7', '17226.40908', '50.200.1.6', '147.32.80.9', 'DNS', '69', 'Standa ...
4	list	7	['8', '17226.40908', '147.32.84.165', '147.32.80.9', 'DNS', '69', 'Sta ...
5	list	7	['9', '17226.40908', '147.32.84.165', '147.32.80.9', 'DNS', '69', 'Sta ...
6	list	7	['10', '17226.40908', '147.32.84.165', '147.32.80.9', 'DNS', '69', 'St ...
7	list	7	['11', '17226.40957', '23.15.86.8', '147.32.84.165', 'DNS', '69', 'Sta ...
8	list	7	['12', '17226.94993', '23.15.86.8', '147.32.84.165', 'DNS', '72', 'Sta ...
9	list	7	['13', '17226.94993', '147.32.80.9', '147.32.84.165', 'DNS', '72', 'St ...
10	list	7	['14', '17227.02023', '191.100.81.3', '147.32.80.9', 'DNS', '72', 'Sta ...
11	list	7	['15', '17227.02024', '191.100.81.3', '147.32.80.9', 'DNS', '72', 'Sta ...
12	list	7	['24', '17227.41045', '147.32.84.165', '147.32.80.9', 'DNS', '69', 'St ...
13	list	7	['25', '17227.41046', '147.32.84.165', '147.32.80.9', 'DNS', '69', 'St ...

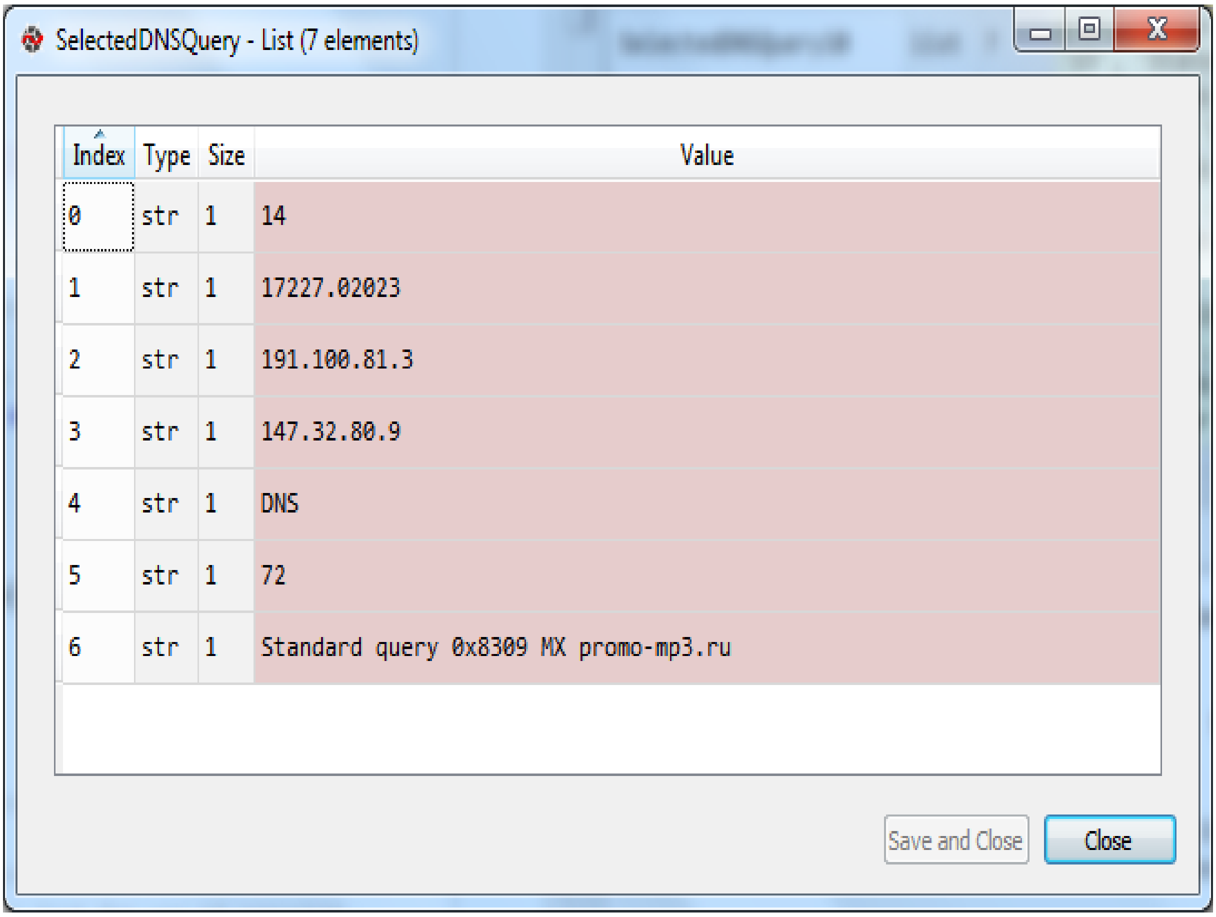
Save and Close Close

Figure 4.4: Selected Protocol List

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

4.2.4 DNS Query

Below is the snap of single DNS query having source IP, DNS Server IP, Protocol and the URL that is required to be accessed by end user:-



Index	Type	Size	Value
0	str	1	14
1	str	1	17227.02023
2	str	1	191.100.81.3
3	str	1	147.32.80.9
4	str	1	DNS
5	str	1	72
6	str	1	Standard query 0x8309 MX promo-mp3.ru

Figure 4.5: A DNS Query

4.2.5 Results

This section shows that results that is provided by the Classification module which classified that either the traffic is a BOTNET CnC traffic or legitimate traffic. Results are given below:-

promo-mp3.ru is Clean Traffic not CnC [Class-1] Source IP: 191.100.81.3 Destination IP: 147.32.80.9 Destination Protocol: DNS

thiswebonline.com is a CnC Attempt of Botnet [Class-2] Source IP: 147.32.80.9 Destination IP: 147.32.80.9 Destination Protocol: DNS

irc.zief.pl is a CnC Attempt of Botnet [Class-2] Source IP: 147.32.84.165 Destination IP: 147.32.80.9 Destination Protocol: DNS

dns4.zief.pl is a CnC Attempt of Botnet [Class-2] Source IP: 147.32.80.9 Destination IP: 147.32.80.9 Destination Protocol: DNS

ii.ebatmoyhuy.com is a CnC Attempt of Botnet [Class-2] Source IP: 54.208.248.114 Destination IP: 147.32.80.9 Destination Protocol: DNS

accessonline-dlbtransfer.com is a CnC Attempt of Botnet [Class-2] Source IP: 147.32.84.165 Destination IP: 147.32.80.9 Destination Protocol: DNS

api.iris.microsoft.com is Clean Traffic not CnC [Class-1] Source IP: 23.15.86.8 Destination IP: 147.32.80.9 Destination Protocol: DNS

aduidc.xyz is a CnC Attempt of Botnet [Class-2] Source IP: 221.154.7.98 Destination IP: 147.32.80.9 Destination Protocol: DNS

pippio.com is Clean Traffic not CnC [Class-1] Source IP: 50.200.1.6 Destination IP: 147.32.80.9 Destination Protocol: DNS

soluxury.co.uk is a CnC Attempt of Botnet [Class-2] Source IP: 201.232.100.96 Destination IP: 147.32.80.9 Destination Protocol: DNS

hotjar.com is Clean Traffic not CnC [Class-1] Source IP: 147.32.84.165 Destination IP: 147.32.80.9 Destination Protocol: DNS

people-pa.googleapis.com is Clean Traffic not CnC [Class-1] Source IP: 147.32.84.165 Destination IP: 147.32.80.9 Destination Protocol: DNS

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

l.betrad.com is Clean Traffic not CnC [Class-1] Source IP: 147.32.84.165 Destination IP: 147.32.80.9 Destination Protocol: DNS

specialistups.com is a CnC Attempt of Botnet [Class-2] Source IP: 91.200.4.8 Destination IP: 147.32.80.9 Destination Protocol: DNS

mulbora.com is a CnC Attempt of Botnet [Class-2] Source IP: 100.50.64.98 Destination IP: 147.32.80.9 Destination Protocol: DNS

Accuracy of the proposed solution has been calculated using following formula

Accuracy =

$$\frac{TP + TN}{TP + TN + FP + FN}$$

All the datasets were passed from the algorithm to find out the accuracy of our detection and prevention model. Results shows 83 Percent accuracy.

4.3 Future Work

For future work, the model that has been proposed in this Paper can be used further with other Machine learning algorithms and protocols as well to check the accuracy and performance of proposed solution with all available options.

References

[1] Ayesha Imran, (2017). SDN Controller Security Issues. University of Jyväskylä, Department of Mathematical Information Technology.

[2] Tariq, F., Baig, S. (2016). Botnet classification using centralized collection of network flow counters in software defined networks. International Journal of Computer Science and Information Security, 14(8), 1075.

[3] Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C. (2012, December). Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In Proceedings of the 28th Annual Computer Security Applications Conference (pp. 129-138). ACM.

[4] Stevanovic, M., Pedersen, J. M. (2014, February). An efficient flow-based botnet detection using supervised machine learning. In 2014 international conference on computing, networking and communications (ICNC) (pp. 797-801). IEEE.

[5] Hadianto, R., Purboyo, T. W. (2018). A Survey Paper on Botnet Attacks and Defenses in Software Defined Networking. International Journal of Applied Engineering Research, 13(1), 483-489.

[6] Tariq, F. Baig, S. (2017). Machine learning based botnet detection in software defined networks. International Journal of Security and Its Applications, 11(11), 1-11.

[7] Letteri, I., Del Rosso, M., Caianiello, P., Cassioli, D. (2018). Performance of Botnet Detection by Neural Networks in Software-Defined Networks. In ITASEC

[8] Feily, M., Shahrestani, A., Ramadass, S. (2009, June). A survey of botnet and botnet detection. In 2009 Third International Conference on Emerging Security Information, Systems and Technologies (pp. 268-273). IEEE.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

[9] Wijesinghe, U., Tupakula, U., Varadharajan, V. (2015, January). An enhanced model for network flow based botnet detection. In Proceedings of the 38th Australasian Computer Science Conference (ACSC 2015) (Vol. 27, p. 30).

[10] Su, S. C., Chen, Y. R., Tsai, S. C., Lin, Y. B. (2018). Detecting p2p botnet in software defined networks. Security and Communication Networks, 2018.